**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

**BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

In re application of:

Ilya Kirnos

Serial No.: 09/866,143

Filed: May 25, 2001

For: MANAGEMENT AND SYNCHRONIZATION APPLICATION FOR NETWORK FILE SYSTEM

Group Art Unit No.: 2143

Examiner: SHIN, Kyung H.

Confirmation No.: 4448

**Mail Stop Appeal Briefs-Patents**
Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

## REPLY BRIEF

Sir:

This Reply Brief is submitted under 37 C.F.R. §1.193(b)(1), in response to Examiner's Answer, mailed September 11, 2007, for which the period of reply runs until November 11, 2007. This reply brief is submitted on the first business day after November 11, 2007, and therefore in a timely manner.

## REMARKS

The status of claims and grounds of rejection are as stated in the Appeal Brief, dated August 2, 2007. The remarks below are made in response to Examiner's arguments in the Answer dated September 11, 2007. These remarks stand in addition to those made in the Appeal Brief, each of which remains valid in view of the Examiner's Answer.

I. THE CITED REFERENCES DO NOT DISCLOSE A COMPARISON FILE WITHIN THE MEANING OF THE CLAIMS.

Claim 1 recites, among other elements:

> recording information about one or more items in a file system to a **comparison file**, wherein the information recorded to the comparison file includes **location information to identify where in the file system the one or more items are located**;

Applicant has repeatedly pointed out that the cited references do not disclose a "comparison file" within the meaning of Claim 1. Until the Examiner's Answer, Examiner had thus far avoided clearly stating what element of the cited references corresponded to a "comparison file." In Examiner's Answer, Examiner explained that Verma's "transaction context" is a "comparison file." Answer at 30. However, as explained below, a transaction context cannot be a comparison file.

### *A. A Transaction Context is an Object, not a "File"*

To say that a transaction context is a "comparison file" ignores the plain and ordinary meaning of the word "file." Verma repeatedly uses the term "object" to refer to a transaction context. *See, e.g.,* Verma at col. 7, line 7. In computer science, the term "object" has a technical meaning, such that a "file" cannot be an "object." For instance, an object is generally understood to be a transient, in-memory structure, whereas a file is stored more persistently on disc. Verma discusses its transaction context in terminology consistent with the technical sense of the word object. *See, e.g.,* Verma at col. 16, lines 10–54.

Nonetheless, the Examiner alleges that, in Verma, "information concerning the changes to a file system is stored in a file" (Answer at 30), and that this supposed "file can be used to synchronize the actual file system." Verma's transaction context object is *not* a file, but a transient object that exists in a computer's memory only for the duration of a transaction. One skilled in the art would readily recognize a number of functional differences between a file and a transient object. For example, the transience of Verma's transaction context would frustrate any attempts to use it to maintain a synchronized file system.

### *B. A Transaction Context does not include "location information to identify where in the file system the one or more items are located."*

Claim 1 specifically requires that the comparison file include "location information to identify where in the file system the one or more items are located." While Applicant has repeatedly explained that Verma's transaction context does not record any file system information at all, *see, e.g.,* Appeal Brief at 10, it is at very least clear that the transaction context does not include **location information**. Nor has Examiner explained how a transaction context includes location information.

The Examiner's only suggestion that Verma tracks location information is the statement that Verma's transaction context somehow "discloses file system transactions, such as the movement of a file to a new location." The Examiner bases this disclosure on Verma, col.2, lines 14–17, which indeed state that Verma's **transactions** can perform "multiple file system operations." However, neither this line nor any line of Verma states that a **transaction context** (not be confused with a transaction itself) actually includes any information about the file system, much less "**location** information" identifying the location of an item "in the file system." The Examiner appears to assume that a transaction context stores such information, when in fact it does not.

Below is a list of all occurrences of the term "transaction context" in Verma:

- Col. 2, lines 25–30: "*For new file creations, the parent directory is marked as transacted, and the application may associate a transaction with a thread/process, whereby file operations by such threads/processes are transacted in the context of the specified transaction. Further, the application may choose to*

*instruct the system (e.g., via an API) that child threads/processes inherit the transaction context. . . .*"

- Col. 7, lines 6–7: "*This call creates a transaction object/context 78 that represents the transaction.*"
- Col. 7, lines 15–19: "*The application's first call to the file system 62 identifies a file, a directory or the application's current thread/process, which may have the transaction context 78 associated therewith. If a transaction context is associated, the file system 62 calls out to the TxF 70.*"
- Col. 8, line 65–67: "*When the flag is set, the system of the present invention automatically includes this file in a transaction context.*"
  [Note that cols. 9 and 11 clarify that including a file in a transaction context is simply an abstract way of saying that the file handle for the file points to the transaction context, and thus operations performed on the file are performed transactionally].
- Col. 9, lines 1–17: "***To this end**, as generally represented in FIG. 3, when a create request 80 comes into the file system (NTFS) 62 via an I/O request packet (IRP), an existing transaction context 78 may be attached to the request by passing a pointer to that context 78, whereby the file may be created/opened as part of the existing transaction context 78. Alternatively, if the pointer to Itransaction pointer is NULL in the CreateFileEx API call, the context is picked up off the thread automatically, as in Microsoft.RTM. Transaction Server (MTS) / Component Object Model (COM) model. The file handle 90 returned in response to a successful create/open request 80 will include a pointer to the transaction context 78. Thereafter, calls made with that handle 90 are recognized via the pointer as having a transaction context associated therewith, from which the relevant transaction is identified, and file system operations using that handle are performed on behalf of the transaction until the transaction is ended.*"
- Col. 9, lines 21–25: "*Also defined is the parameter that may point to a transaction context object, LPUNKNOWN punkTransaction, wherein if NULL, as described above, the object is picked up from the current MTS/COM context.*"

- Col. 10, lines 4–7: "*If a particular CreateFileEx specifies a non-null ITransaction object pointer, that object is used for the transaction context 78, otherwise the MTS transaction object is picked up off the thread.*"
- Col. 11, lines 1–12: "*As shown in FIG. 4, for file operations other than create/open, the application 60 provides the handle 90 to the file system, e.g., via an API call 92 requesting a file read operation, whereby via the transaction context pointer therein, the file system can locate the transaction context. Note that TxF 70 may have to enlist the transaction, as described above. Because of the transaction context pointed to in the file handle 90, the file system knows that this operation is included in a transaction, as well as the identifier of the particular associated transaction.* ***Including a file in the transaction context means operations on the file will be transacted****, including reads, writes, file creation and deletion.*"
- Col. 16, lines 14–26: "*If a transaction opens a file for read/write, then TxF requires one structure for the file, one per stream, and one for the stream version to store its per-transaction context, as represented in FIG. 9. The data structures for this open are represented in FIG. 9, where "File Object" is the object mapped by the user's file handle, "FCB" is the NTFS file control block, "SCB" is the NTFS stream control block for the specific stream opened, "NP SCB" is the non-paged stream control block used primarily to hold the section object pointers for file mapping, and "CCB" is the per-FileObject context structure.*"
  [Assuming that CCB is a transaction context, an assumption which is by no means certain, note that FIG. 9 specifically shows that the only thing the transaction context (CCB) points to is the TxFFO, which in turn points to the TxFSCB, which is the "anchor for the undo data." In other words, one must follow two object pointers from the CCB before reaching anything that even resembles file information. This hardly constitutes including location information about a file within the CCB.]

None of these references suggests that location information is actually included in a transaction context. In fact, the only thing that that is certain about the contents of Verma's

transaction context is that, from those contents, "a relevant transaction is identified." Verma at col. 9 line 15. Achieving this goal does not actually require the transaction context to itself contain "location information." In fact, from FIG. 9, it appears that this goal is achieved simply by pointing to a transaction object.

### *C. Information in a Transaction Context cannot be compared to a working version to determine if items have "been moved to a new location in [a] working version."*

Claim 1 also requires that the location information in the comparison file be compared to a "working version to determine if . . . items ha[ve] been moved to a new location in the working version." Yet Examiner appears to believe that Verma's transaction context includes a "working version of management information." Answer at 33. If such is the case, it would be impossible for a transaction context to function as a comparison file—a comparison file does not include information about the working version, rather, information in a comparison file is compared to the working version.

For at least the reasons discussed above, as well as reasons discussed in the Appeal Brief, a transaction context cannot be considered a comparison file. Thus, Verma does not disclose a comparison file as required by Claim 1. It is not alleged that any other reference discloses a comparison file, and in fact the cited references neither anticipate nor render obvious at least this aspect of Claim 1. Claim 1 is patentable over the cited references for at least this reason.

Furthermore, each of independent claims 12, 25, 31, 76, 87, 99, and 105 require a comparison file. These claims are likewise patentable over the cited references for this same reason.

## II. THE CITED REFERENCES DO NOT DISCLOSE A "WORKING VERSION OF THE FILE SYSTEM"

Claim 1 requires, among other elements:

> generating a working version of a portion of the file system, the working version including at least one or more working items that correspond to the one or more items located in the file system;

Examiner has suggested that Verma, col. 16, lines 39–44 disclose this feature. These lines read:

> FIG. 10 thus represents one read/write open by transaction t3 modifying the current version of the file, one read-only open by transaction t2 accessing the most recent committed version of the file, and another read-only open by transaction t1 accessing an earlier committed version.

Thus far, the Examiner has avoided stating how this passage from Verma discloses a "working version of a portion of the file system." In fact, this passage of Verma fails to disclose a "working version of a portion of the file system" for at least the following reasons.

### *A. Verma does not disclose a "working item"*

As required by Claim 1, a "working version of a portion of the file system" requires "one or more working items." The Examiner alleges that the "first and second operational states" in the above quoted passage of Verma correspond to "working items."

The cited passage reveals nothing more than a mechanism for maintaining an "undo" version of a file to which a transaction is writing. Examiner apparently analogizes the transacted version of the file to a "working item" because it corresponds to an original version of the file in the file system (i.e. the undo version).

This analogy ignores the fact that other clauses of Claim 1 require working items to "ha[ve] been moved to a new location in the working version." Apart from the fact that there is no working version of the file system in which to move the transacted version of the file, the transacted version cannot be a working item because it cannot be "moved" anywhere in the file system relative to the original undo version. Both the transacted version and the "undo" version still occupy the same location in the file system—as manifest by the fact that other transactions read both versions of the file by the same name. From a file system standpoint, the versions *do not even exist*. See, e.g., Verma at col. 16, lines 45–46 ("NTFS is unaware of the file versions.").

Thus, Verma fails to disclose a "working item" as required by Claim 1.

### *B. Verma does not disclose a "working version"*

Furthermore, the Examiner has never specifically stated what element of Verma corresponds to a "working version of a portion of the file system." Rather, the Examiner appears to assume that because, as he alleges, Verma discloses "working items," Verma also discloses a "working version of a portion of the file system."

The Examiner's logic is in error. The claim specifically requires "a working version of the file system," not just "one or more working items." The term "working version of the file system," has a plain and ordinary meaning that would be obvious to one skilled in the art. To suggest that something is a working version of a file system just because it contains a working item ignores this plain and ordinary meaning.

Nor does the existence of "one or more working items" in Verma imply that there is a "working version of a portion of the file system." Indeed, **there is only one version of a file system** in Verma. The cited portion of Verma merely discloses keeping multiple versions of a file in memory for the duration of a transaction, so that the original version of the file may be restored to the file system in the event of a transaction failure. There is nothing in Verma that suggests something like a "working version of a portion of the file system."

### *C. Verma teaches away from a "working version"*

Claim 1 also makes it clear that a key aspect of a "working version" is that the working version is used in "a synchronization event." Verma, however, has absolutely nothing to do with synchronizing multiple versions of a file system. The only point in common between Verma's techniques for a single, transactional-based file system and Applicant's techniques for synchronizing multiple versions of a file system is that both involve a file system. No aspect of Verma suggests to one skilled in the art the novel features of Applicant's invention. In fact, Verma teaches away from "versioning" in the synchronization sense by *explicitly stating* that Verma's "versioning" "should not be confused with persistent versioning such as in a source code control system." Verma at col. 12, lines 10–12.

Thus, not only does Verma fail to teach "generating a working version of a portion of a file system," Verma specifically disclaims any relevance to the type of versioning required for synchronization.

For at least the reasons discussed above, as well as reasons discussed in previous Responses to Office Actions, Verma does not disclose "generating a working version of a portion of the file system" as required by Claim 1. It is not alleged that any other reference discloses "generating a working version of a portion of the file system" within the meaning of Claim 1. Thus, Examiner has failed to make a prima facie case for an obviousness rejection of Claim 1 under 35 U.S.C. § 103(a).

Each of independent claims 12, 25, 31, 76, 87, 99, and 105 likewise require "a working version of a portion of the file system," and Examiner has likewise failed to make a prima facie case for an obviousness rejection of these Claims under 35 U.S.C. § 103(a).

## III. THE CITED REFERENCES DOES NOT DISCLOSE "COMPARING . . . INFORMATION IN THE COMPARISON FILE TO THE WORKING VERSION"

Claim 1 requires, among other elements:

> upon a synchronization event, **comparing the location information** for the one or more items **in the comparison file to the working version** to determine if any of the corresponding one or more working items has been moved to a new location in the working version.

Even if the cited references did disclose a comparison file and a working version, Examiner has not explained how the cited references teach **to actually perform the step of comparing** information in the comparison file to the working version. More specifically, the Examiner has not explained why it would have been obvious to compare information in Verma's transaction context with his alleged working versions.

In fact, it would not have been obvious to have made such a comparison because: (1) according the Examiner (as discussed in section I.C above) the transaction context already contains information about the alleged working version; and (2) there would have been no motivation to compare the transaction context to the alleged working version.

Examiner suggests that Verma discloses this aspect of Claim 1 in view of Bailey. However, Bailey also does not teach to compare a comparison file to a working version. In fact, Bailey teaches to copy each file in a file directory to another directory. If a file is identified as an

anti-file, a same named file in the other directory may be deleted. *See* Bailey at col. 5, lines 59–65. This aspect of Bailey neither teaches nor renders obvious comparing *location information in a comparison file* to a working version. *See, e.g.,* Appeal Brief at 13.

For at least the reasons discussed above, as well as reasons discussed in the Appeal Brief, the cited references do not disclose "**comparing the location information** for the one or more items **in the comparison file to the working version**" as required by Claim 1. Claim 1 is patentable over the cited references for at least this reason.

Furthermore, each of independent claims 12, 25, 31, 76, 87, 99, and 105 require a similar comparison, though the comparison may be of any information in the comparison file, as opposed to having to include "location information." Nonetheless, these claims are likewise patentable over the cited references for the above reason.

## IV. THE CITED REFERENCES DOES NOT DISCLOSE "DETERMIN[ING] IF ANY . . . WORKING ITEM[] HAS BEEN MOVED IN THE WORKING VERSION."

Claim 1 requires, among other elements:

> upon a synchronization event, comparing the location information for the one or more items in the comparison file to the working version to **determine if any of the corresponding one or more working items has been moved to a new location in the working version**.

The Examiner's Answer suggested that either Verma or Bailey disclosed "determining if any . . . working item[] has been moved in the working version." This is first of all impossible because, as discussed in section II above, Verma discloses no "working item" that may be moved and no "working version" in which a working item may be moved. However, even if the cited references did disclose a comparison file and a working version, Examiner has not explained how the cited references teach **to actually perform the step of determining** if a working item "has been moved to a new location in the working version."

In fact, Verma and Bailey neither teach nor render obvious such a determination. Verma, for example, only discloses rolling back a transaction if it cannot successfully commit. Verma at col. 16, lines 39–44. Rolling-back a transaction involves shifting pointers back to the undo

version upon determining that a transaction has failed. It does not require determining if an item has been moved. Bailey, on the other hand, merely contemplates determining if a file has been removed from a location. Bailey neither contemplates nor allows for any determination of an item having been moved to a location in a working version.

The Examiner's Answer suggests that the mere alleged "**capability**" in Verma and Bailey "to compare directory information to determine file system information updates between two entities," Answer at 32, is enough to have disclosed "determin[ing] if any . . . working item[] has been moved in the working version." However, this capability is not a sufficient enough basis for a determination of whether an item has been moved as opposed to simply deleted. Examiner overlooks that a key aspect of Claim 1 is that such a determination must at least be based on a comparison file. *See, e.g.,* Appeal Remarks at 13.

Furthermore, a combination of two references cannot be said to have made an invention obvious to one skilled in the art just because one skilled in the art might be *capable* of practicing the inventive method with the combination. Simply put, it would not have been obvious to one skilled in the art, having merged Verma's transactional file system with Bailey's synchronization system based on anti-files, to, upon a synchronization event, use any feature of either Verma or Bailey to determine if a file had been moved in a working version. This is true for a variety of reasons, such as, for example, the fact that such a determination would have been impossible. *See, e.g.,* Appeal Remarks at 11.

For at least the reasons discussed above, as well as reasons discussed in the Appeal Brief, the cited references do not disclose "determining if any . . . working item[] has been moved in the working version" as required by Claim 1. Claim 1 is patentable over the cited references for at least this reason.

## V. THE CITED REFERENCES DO NOT DISCLOSE "PERSISTENTLY MAINTAINING THE WORKING VERSION"

Claim 1 requires, among other elements, "persistently maintaining the working version." The Examiner alleges that this element of Claim 1 may be found in Rudoff at col. 6, lines 33–40. Applicant has repeatedly explained that Rudoff does not, in fact, have anything to do with a working version of a file system. Rather, Rudoff merely contemplates maintaining versions of

file in the same location in the same file system. *See, e.g.,* Appeal Brief at 13. Examiner has failed to acknowledge Applicant's remarks regarding Rudoff.

For at least the reasons previously explained in the Appeal Brief, Rudoff does not disclose "persistently maintaining the working version." Thus, Examiner has failed to make a prima facie case for an obviousness rejection of Claim 1 under 35 U.S.C. § 103(a).

Each of independent claims 12, 25, 31, 76, 87, 99, and 105 likewise require "persistently maintaining the working version." Thus, Examiner has failed to make a prima facie case for an obviousness rejection of these claims as well.

## VI. THERE WOULD HAVE BEEN NO MOTIVATION TO COMBINE

The Examiner has repeatedly avoided responding to Applicant's arguments that there would have been no motivation to combine Verma, Bailey, and Rudoff. *See, e.g.,* Appeal Brief at 13–15. The only relationship between these references is that they involve a file system.

Particularly frustrating to Applicant has been Examiner's continued reliance on Verma when, as stated in section II.C above, Verma teaches away from "versioning" in the synchronization sense by *explicitly stating* that Verma's "versioning" "should not be confused with persistent versioning such as in a source code control system." Verma at col. 12, lines 10–12. The inconsistencies highlighted in sections I and II above further underscore the contrived nature of this combination of Verma, Bailey, and Rudoff.

Even if Rudoff taught "persistent versioning" as required by the Claims, Verma would nonetheless be inoperative with Rudoff because Verma teaches away from persistent versioning. *See supra* at § II.C. Thus, not only would there have been no motivation to combine Rudoff with Verma; *Rudoff could not have been used with Verma's versioning.*

Thus, as discussed in the Appeal Brief, there is no motivation to combine Verma, Bailey, and Rudoff. The hindsight combination of these unrelated references cannot be said to render any of the rejected claims obvious under 35 U.S.C. § 103(a).

## VII. THE CITED REFERENCES DO NOT DISCLOSE "DETERMIN[ING] IF A FIRST WORKING ITEM . . . WAS COPIED FROM A SECOND WORKING ITEM."

Claim 12 features, among other elements, "determining if a first working item in the working version was copied from a second working item." As with Claim 1, Examiner gives no explanation of how either Verma or Bailey teach **the actual step of determining**. *See supra* at § IV.

Also, the Examiner suggests that "a comparison would indicate that the two working items were identical (copies)." Examiner appears to believe that, just because two items are identical, one item was copied from the other. This is not necessarily true. A file system may possess many identical files that are independently generated.

Thus, the cited references could not determine, solely on the basis of files being identical, that a first item was copied from a second item. Nor do the cited references disclose any other method for making such a determination. For at least these reason, Claim 12 is patentable over the cited references.

## VIII. THE CITED REFERENCES DO NOT DISCLOSE "DETERMINING IF A COMPOUND OPERATION WAS PERFORMED ON THE FIRST WORKING ITEM"

Claim 31 requires, among other elements, "determining if a compound operation was performed on the first working item." As with Claim 1, Examiner gives no explanation that either Verma or Bailey teach **the actual step of determining**. *See supra* at § IV.

Furthermore, the Examiner did not respond to Applicant's explanation that neither Verma nor Bailey discloses this feature. *See* Appeal Brief at 17–18. For at least the reason that the cited references do not disclose "determining if a compound operation was performed on the first working item," Claim 31 is patentable over the cited references.

## IX. DEPENDENT CLAIMS 2–11, 13–24, 26–30, 32–33, 77–86, 88, 90–98, 100–104, AND 106–107 ARE SEPARATELY PATENTABLE

Applicant has repeatedly argued that each of dependent claims 2–11, 13–24, 26–30, 32–33, 77–86, 88, 90–98, 100–104, and 106–107 are patentable over the cited references. *See, e.g.,*

Appeal Brief at 18–22. In that Appeal Brief, Applicant presented several reasons why certain of these claims were separately patentable over the cited references.

However, in the examination of these dependent claims, Examiner has repeatedly ignored several key aspects of the dependent claims, such as Claim 7's "**using** the creation time to **determine** if [a] working item[] has been deleted," Claim 10's "**determining** if . . . working items were edited . . . **using** the modification time," and Claim 11's "**determining** if a subsequent modification time . . . is different than the recorded modification time." Examiner has presented no evidence that making such determinations or using creation or modification times in this manner would actually be obvious to one skilled in the art.

Rather, the Examiner simply assumes that it would have been obvious to perform these steps because the cited references possess, in Examiner's opinion, the tools used to perform these steps. Applicant readily admits that certain tools used in his invention exist in the cited references—e.g., a modification time. To the extent that these certain tools do exist in the cited references, it is Applicant's particular use of these tools in the Claims that is novel, not the tools themselves.

With regards to each of Claims 7, 10, and 11, the Examiner blanketly states that it would have been obvious use the creation time or modification time to make the claimed determinations. *See, e.g.,* Answer at 18–20. Examiner does not explain why it would have been obvious to use the creation time or modification time in this manner. Nor does Examiner explain why, if it were so obvious to use creation times and modification times in this manner, Bailey resorts to a much more complicated scheme of anti-files to detect the deletion of a working item, which detection Claim 7 ultimately accomplishes.

In fact, the cited references do not disclose the above listed features of Claims 7, 10, and 11. Claims 7, 10, and 11 are patentable over the cited references for at least this reason. Furthermore, the cited references do not disclose many other features of the dependent claims 2–11, 13–24, 26–30, 32–33, 77–86, 88, 90–98, 100–104, and 106–107. However, these reasons have either already been argued, or, for the sake of brevity, are not argued at this time.

## CONCLUSION AND PRAYER FOR RELIEF

As explained in both this Reply Brief and the Appeal Brief, the rejections to Claims 1-33, 76-88, and 90-107 made under 35 U.S.C. § 103(a) lack the requisite factual and legal basis. The Appellant respectfully submits that the imposed rejections to Claims 1-33, 76-88, and 90-107 are not viable and respectfully solicit the Honorable Board to reverse each of the imposed rejections.

The Director is hereby authorized to charge any additional fee(s) or underpayments of fee(s) under 37 C.F.R. 1.16 and 1.17 to Deposit Account Number 50-1302.

Respectfully submitted,
HICKMAN PALERMO TRUONG & BECKER LLP

/KarlTRees#58983/
Karl T. Rees
Reg. No. 58,983

2055 Gateway Place, Suite 550
San Jose, CA 95110-1089
(408) 414-1080
Date: November 13, 2007
Facsimile: (408) 414-1076